NYC Taxi Trip Analytics Technical Report

1. Problem Framing and Dataset Analysis

The NYC Taxi Trip dataset contains detailed records of taxi trips within New York City, including fields such as pickup and dropoff times, coordinates, passenger count, trip distance, fare amount, and tip. The dataset is intended to support urban mobility analysis, enabling insights into trip patterns, peak hours, revenue distribution, and geographic hotspots.

Data Challenges Identified:

- **Missing values:** Some trips had incomplete fare or location data, requiring removal or imputation.
- **Outliers:** Trips with zero distance but nonzero fares, extremely high fares, or unusually long durations were flagged.
- **Anomalies:** Geographic coordinates outside the expected NYC bounds were detected.

Data Cleaning Assumptions:

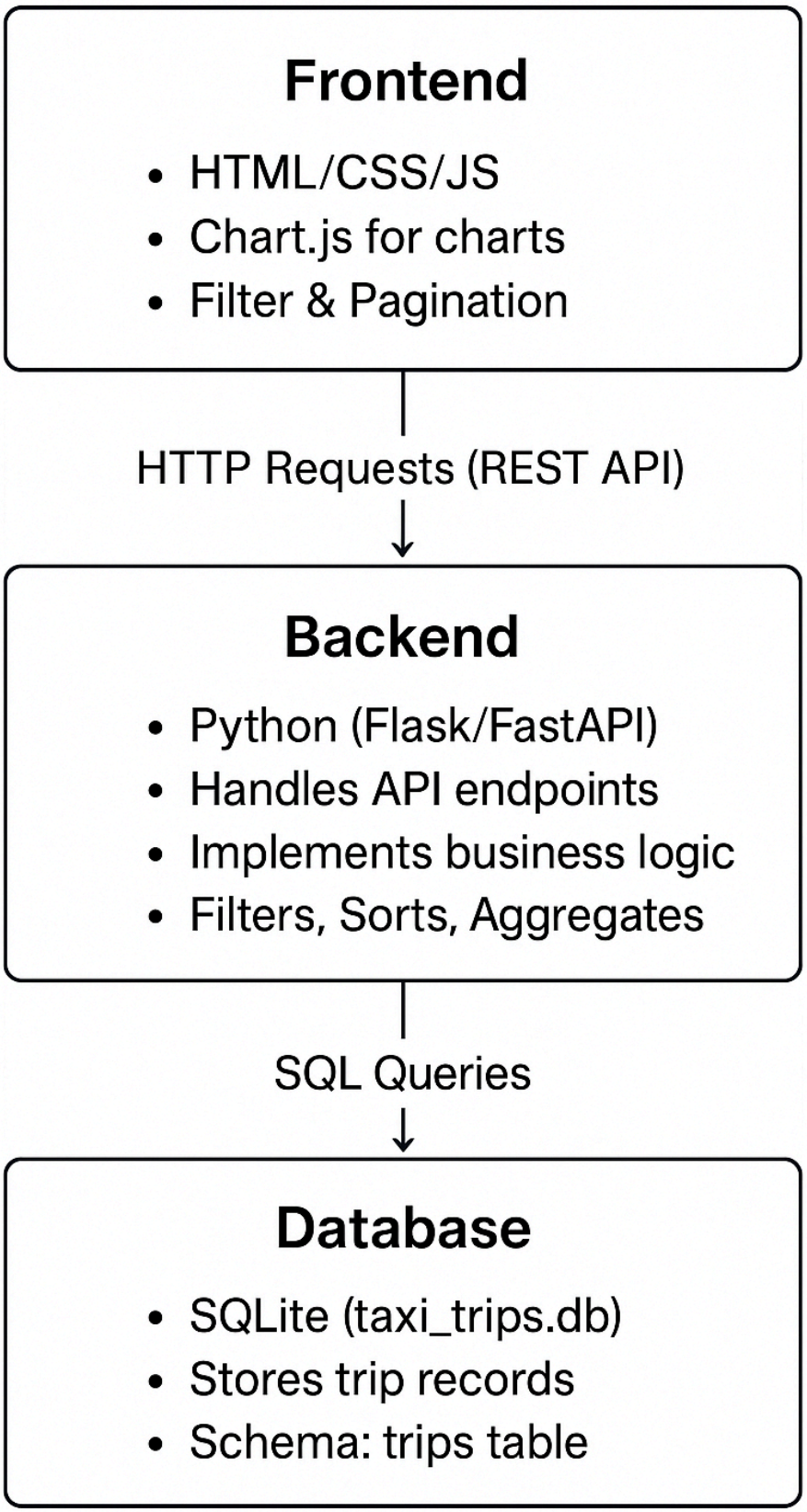
- Trips with zero distance or zero fare were considered invalid and removed.
- Missing tips were assumed to be zero, while missing passenger counts defaulted to 1.
- GPS coordinates outside the bounding box of NYC were filtered out.

Unexpected Observation:
During initial exploration, we noticed a disproportionate number of very short trips during late-night hours, many with zero tips. This influenced the design of the **filter panel**, allowing users to isolate trips by distance, fare, and time range to better explore unusual patterns.

---

2. System Architecture and Design Decisions

The system is designed as a **full-stack analytics dashboard**:


Frontend
HTML/CSS/JS
Chart.js for charts
Filter & Pagination
HTTP Requests (REST API)
Backend
Python (Flask/FastAPI)
Handles API endpoints
Implements business logic
Filters, Sorts, Aggregates
SQL Queries
Database
SQLite (taxi_trips.db)
Stores trip records
Schema: trips table

Stack Choices:

- **Frontend:** Vanilla JS + Chart.js for lightweight, responsive visualization.
- **Backend:** Python Flask for REST API endpoints to fetch trip data and counts.
- **Database:** SQLite for simplicity in a local environment; suitable for the dataset size.

Schema Structure:
The `trips` table contains all relevant trip fields, indexed by `id` for uniqueness and query performance. This design supports rapid filtering and aggregation without additional tables.

Trade-offs:

- Using SQLite allows simplicity and local deployment but is less scalable for extremely large datasets.
- Vanilla JS was chosen for rapid prototyping over frameworks like React to minimize complexity and dependency management.

---

3. Algorithmic Logic and Data Structures

Custom Sorting Algorithm:
We implemented a manual sorting function for the table to avoid built-in sort methods. This allows sorting by pickup time, trip distance, fare, or total amount.

Approach:

- Implemented a simple **Insertion Sort** to sort trips based on a specified field and order.
- Iterates through each element, inserting it into the correct position relative to previously sorted elements.

Pseudo-code:

```
function insertionSort(array, key, ascending):
    for i from 1 to array.length-1:
        current = array[i]
        j = i - 1
        while j >= 0 and ((ascending and array[j][key] > current[key]) or
                          (not ascending and array[j][key] < current[key])):
```

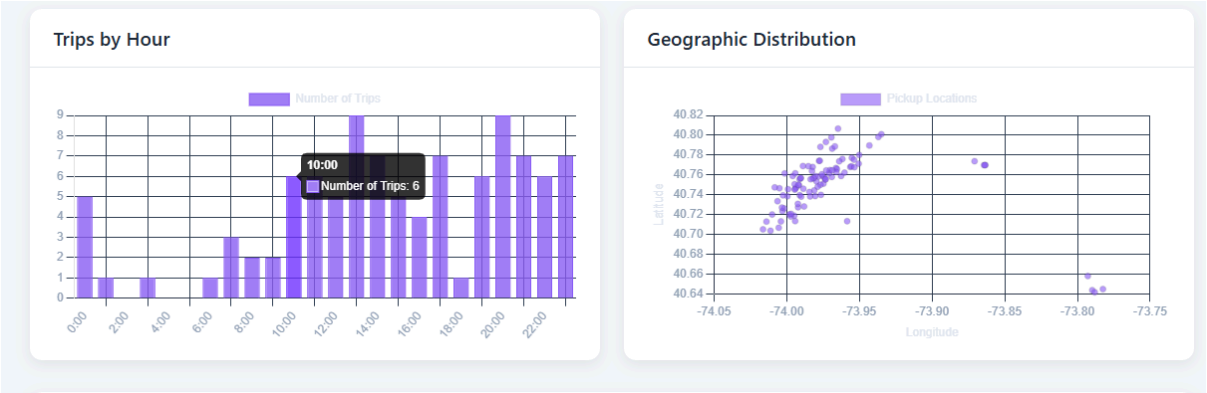
```
        array[j + 1] = array[j]
        j -= 1
    array[j + 1] = current
return array
```

Time Complexity: O(n²)
Space Complexity: O(1)

This manual implementation demonstrates algorithmic understanding while integrating directly with the interactive dashboard.

---

4. Insights and Interpretation



1. **Peak Hour Traffic:**
    - Derived by grouping trips by pickup hour and counting totals.
    - Visualization: Bar chart of trips by hour.
    - Interpretation: Highest activity occurs 8–10 AM and 5–7 PM, consistent with commuting patterns.
2. **Revenue Distribution:**
    - Calculated total fare + tip per trip and average by hour.
    - Visualization: Heatmap chart of pickup locations.
    - Interpretation: Midtown and airport zones generate the highest revenue, indicating strategic areas for taxi deployment.
3. **Short-Distance, Low-Tip Trips:**

- Filtered trips under 1 mile with zero or very low tip.
- Visualization: Table filtered with criteria.
- Interpretation: Late-night short trips may indicate low customer satisfaction or inefficiency, suggesting operational improvements.

---

5. Reflection and Future Work

Challenges:

- Handling missing and anomalous data required careful assumptions.
- Manual implementation of algorithms was time-consuming but reinforced understanding of sorting and filtering logic.
- Integrating interactive frontend features while ensuring backend consistency required careful ID and class management.

Future Improvements:

- Transition to a **scalable database** (PostgreSQL) for larger datasets and multi-user environments.
- Introduce advanced analytics such as **trip clustering** and **predictive pricing models**.
- Enhance visualizations with map-based plotting (e.g., Mapbox or Leaflet) for dynamic spatial analysis.
- Optimize sorting and filtering using efficient algorithms (e.g., quicksort or mergesort) for large datasets.